Black-Box Optimization Revisited: Improving Algorithm Selection Wizards through Massive Benchmarking

Laurent Meunier, Herilalaina Rakotoarison, Pak Kan Wong, Baptiste Roziere, Jeremy Rapin, Olivier Teytaud, Antoine Moreau, Carola Doerr

Black-Box Optimization Revisited: Improving Algorithm Selection Wizards through Massive Benchmarking

Laurent Meunier[1]*, Herilalaina Rakotoarison[3]*, Pak Kan Wong[2]*,
Baptiste Roziere[1], Jeremy Rapin[1],
Olivier Teytaud[1], Antoine Moreau[4], Carola Doerr[5]

[1]Facebook AI Research, Paris, France
[2]The Chinese University of Hong Kong, Hong Kong
[3]TAU, LRI - INRIA - Université Paris-Saclay, Orsay, France
[4]Université Clermont Auvergne, CNRS, SIGMA Clermont, Institut Pascal, Clermont-Ferrand, France
[5]Sorbonne Université, CNRS, LIP6, Paris, France

Abstract

Existing studies in black-box optimization for machine learning suffer from low generalizability, caused by a typically selective choice of problem instances used for training and testing different optimization algorithms. Among other issues, this practice promotes overfitting and poor-performing user guidelines. To address this shortcoming, we propose in this work a benchmark suite, OptimSuite, which covers a broad range of black-box optimization problems, ranging from academic benchmarks to real-world applications, from discrete over numerical to mixed-integer problems, from small to very large-scale problems, from noisy over dynamic to static problems, etc. We demonstrate the advantages of such a broad collection by deriving from it Automated Black Box Optimizer (ABBO), a general-purpose algorithm selection wizard. Using three different types of algorithm selection techniques, ABBO achieves competitive performance on all benchmark suites. It significantly outperforms previous state of the art on some of them, including YABBOB and LSGO. ABBO relies on many high-quality base components. Its excellent performance is obtained without any task-specific parametrization.

The OptimSuite benchmark collection, the ABBO wizard and its base solvers have all been merged into the open-source Nevergrad platform, where they are available for reproducible research.

1 Introduction: State of the Art

Many real-world optimization challenges are black-box problems; i.e., instead of having an explicit problem formulation, they can only be accessed through the evaluation of solution candidates. These evaluations often require simulations or even physical experiments. Black-box optimization methods are particularly widespread in machine learning [SGZ+16, WFT20], to the point that it is considered a key research area of artificial intelligence. Black-box optimization algorithms are typically easy to implement and easy to adjust to different problem types. To achieve peak performance, however, proper algorithm selection and configuration are key, since black-box optimization algorithms have complementary strengths and weaknesses [Ric76, Smi09, Kot14, BKK+16, KT18, KHNT18]. But whereas automated algorithm selection has become standard in SAT solving [XHHLB08] and AI planning [VHCM15], a manual selection and configuration of the algorithms is still predominant in the broader black-box optimization context. To reduce the bias inherent to such manual choices, and to support the automation of algorithm selection and configuration, sound comparisons of the different black-box optimization approaches are needed. Existing benchmarking suites, however, are rather selective in the problems they cover. This leads to specialized algorithm frameworks whose performance suffer from poor generalizability. Addressing this flaw in black-box optimization,

*Equal contribution

we present a unified benchmark collection which covers a previously unseen breadth of problem instances. We use this collection to develop a high-performing algorithm selection wizard, ABBO. ABBO uses high-level problem characteristics to select one or several algorithms, which are run for the allocated budget of function evaluations. Originally derived from a subset of the available benchmark collection, in particular YABBOB, the excellent performance of ABBO generalizes across almost all settings of our broad benchmark suite. Originally implemented as a fork of Nevergrad [RT18], our benchmark collection OptimSuite, the ABBO wizard, and its base solvers have all been merged into the main Nevergrad master, where all the components and all performance data is available open source. Nevergrad automatically reruns all algorithms at certain time intervals and makes all data available on the public dashboard [RT20]. Note that ABBO is called NGOpt8 in the main Nevergrad master, to allow for better version control.

In summary, our contributions are as follows:

(1) OptimSuite Benchmark Collection: OptimSuite combines several contributions that recently led to improved reliability and generalizability of black-box optimization benchmarking, among them LSGO [LTO+13], YABBOB [HAFR09, LMP+20], Pyomo [HLW+17], MLDA [GS18], and MuJoCo [TET12, MGR18], and others (novelty discussed in Section 2).

(2) Algorithm Selection Wizard ABBO: Our algorithm selection technique, ABBO, can be seen as an extension of the Shiwa wizard presented in [LMP+20]. It uses three types of selection techniques: *passive algorithm selection* (choosing an algorithm as a function of a priori available features [BS04, LMP+20]), *active algorithm selection* (a bet-and-run strategy which runs several algorithms for some time and stops all but the strongest [MBT+11, PA12, FM14, ME13, MSKH15, CLRT16, KHNT18]), and *chaining* (running several algorithms in turn, in an a priori defined order [MLH09]). Our wizard combines, among others, algorithms suggested in [VGO+19, HO03, SP97, Pow64, Pow94, LMP+20, HB16, Art15, DLMN17, DDL19, DL16]. Another core contribution of our work is a sound comparison of our wizard to Shiwa, and to the long list of algorithms available in Nevergrad.

Algorithm 1 High-level overview of ABBO. Selection rules are followed in this order, first match applied. d = dimension, budget b = number of evaluations. Details of ABBO are available in the Nevergrad platform [RT18], where ABBO is listed as NGOpt8.

Case	Choice
Discrete decision variables only	
Noisy optimization with categorical variables	Genetic algorithm mixed with bandits [HMI09, LMP+20].
alphabets of size < 5, sequential evaluations	$(1+1)$-Evolutionary Alg. with linearly decreasing stepsize
alphabets of size < 5, parallel case	Adaptive $(1+1)$-Evolutionary Alg. [DDL19].
Other discrete cases with finite alphabets	Convert to the continuous case using SoftMax as in [LMP+20] and apply CMandAS2 [RDP+19]
Presence of infinite discrete domains	FastGA [DLMN17]
Numerical decision variables only, evaluations are subject to noise	
$d > 100$	progressive optimization as in [Ber16].
$d \leq 30$	TBPSA [HB16]
$b > 100$	sequential quadratic programming
Other cases	TBPSA [HB16]
Numerical decision variables only, high degree of parallelism	
Parallelism $> b/2$ or $b < d$	MetaTuneRecentering [MDRT20]
Parallelism $> b/5$, $d < 5$, and $b < 100$	DiagonalCMA-ES [RH08]
Parallelism $> b/5$, $d < 5$, and $b < 500$	Chaining of DiagonalCMA-ES (100 asks), then CMA-ES+meta-model [AST05]
Parallelism $> b/5$, other cases	NaiveTBPSA as in [CT20]
Numerical decision variables only, sequential evaluations	
$b > 6\,000$ and $d > 7$	Chaining of CMA-ES and Powell, half budget each.
$b < 30d$ and $d > 30$	$(1+1)$-Evol. Strategy w/ 1/5-th rule [Rec73]
$d < 5$ and $b < 30d$	CMA-ES + meta-model [AST05]
$b < 30d$	Cobyla [Pow94]
For all other cases and all details, please refer to the source code	

2 Sound Black-Box Optimization Benchmarking

We summarize desirable features and common shortcomings of black-box optimization benchmarks and discuss how OptimSuite addresses these.

Generalization. The most obvious issue in terms of generalization is the statistical one: we need sufficiently many experiments for conducting valid statistical tests and for evaluating the robustness of algorithms' performance. This, however, is probably not the main issue. A biased benchmark, excluding large parts of the industrial needs, leads to biased conclusions, no matter how many experiments we perform. Inspired by [RRSS18] in the case of image classification, and similar to the spirit of cross-validation for supervised learning, we use a much broader collection of benchmark problems for evaluating algorithms in an unbiased manner. Another subtle issue in terms of generalization is the case of instance-based choices of (hyper-)parameters: an experimenter modifying the algorithm or its parameters specifically for each instance can easily improve results by a vast margin. In this paper, we consider that only the following problem properties are known in advance (and can hence be used for algorithm selection and configuration): the dimension of the domain, the type and range of each variable, their order, the presence of noise (but not its intensity), the budget, the degree of parallelism (i.e., number of solution candidates that can be evaluated simultaneously). To mitigate the common risk of over-tuning, we evaluate algorithms on a broad range of problems, from academic benchmark problems to real-world applications. Each algorithm runs on all benchmarks without any change or task-specific tuning.

Use the ask, tell, and recommend pattern. Formalizing the concept of numerical optimization is typically made through the formalism of oracles or parallel oracles [Rog87]. A recent trend is the adoption of the ask-and-tell format [CHP$^+$10]. The bandit literature pointed out that we should distinguish *ask*, *tell*, and *recommend*: the way we choose a point for gathering more information is not necessarily close to the way we choose an approximation of the optimum [BMS11, Cou12b, DT13]. We adopt the following framework: given an objective function f and an *optimizer*, for $i \in \{1, \ldots, T\}$, do $x \leftarrow \mathit{optimizer.ask}$ and $\mathit{optimizer.tell}(x, f(x))$. Then, evaluate the performance with $f(\mathit{optimizer.recommend})$. The requirement of a recommend method distinct from the ask is critical in noisy optimization. A debate pointed out some shortcomings in the the noisy counterpart of BBOB [AH09] which was assuming that $\mathit{ask} = \mathit{recommend}$: [Bey12a, Bey12b, Cou12a] have shown that in the noisy case, this difference was particularly critical, and a framework should allow algorithms to "recommend" differently than they "ask". A related issue is that a run with budget T is not necessarily close to the truncation of a run in budget $10T$.

Translation-invariance. Zero frequently plays a special role in optimization. For example, complexity penalizations often "push" towards zero. In control, numbers far from zero are often more likely to lead to bang-bang solutions (and therefore extract zero signal, leading to a needle-in-the-haystack optimization situation), in particular with neural networks. In one-shot optimization, [CCD$^+$19, MDRT20] have shown how much focusing to the center is a good idea in particular in high-dimension. Our experiments in control confirm that the scale of the optimization search is critical, and explains the misleading results observed in some optimization papers (Section 4.2). In artificial experiments, several classical test functions have their optimum in zero. To avoid misleading conclusions, it is now a standard procedure, advocated in particular in [HAFR09], to randomly translate the objective functions. This is unfortunately not always applied.

Rotation and symmetrization. Some optimization methods may perform well on separable objective functions but degrade significantly in optimizing non-separable functions. If the dimension of a separable objective function is d, these methods can reduce the objective function into d one-dimensional optimization processes [Sal96]. Therefore, [HAFR09, HRM$^+$11] have insisted that objective functions should be rotated to generate more difficult non-separable objective functions. However, [BGK$^+$17] pointed out the importance of dummy variables, which are not invariant per rotation; and [Hol75] and more generally the genetic algorithms literature insist that rotation does not always makes sense – we lose some properties of a real-world objective function, and in some real-world scenarios rotating would, e.g., mix temperature, distance and electric intensity. Permutating the order of variables is also risky, as their order is sometimes critical - k-point crossovers a la Holland [Hol75] typically assume some order of variables, which would be broken. Also, users sometimes rank variables with the most important first – and some optimization methods do take care of this [CCD$^+$19]. In OptimSuite, we do include rotations, but include both cases, rotated or not.

Table 1: Properties of selected benchmark collections (details in appendix). "+" means that the feature is present, "-" that the feature is missing, and an empty case that it is not applicable. Far-optimum refers to problems with optimum far from the center or on the side of the domain; such benchmarks test the ability of optimization algorithms to answer promptly to a bad initialization [CAH12]. "Translations" applies only to artificial benchmarks. A "+" in rows "Multimodal", "symmetrization", and "real-world" does not imply that *all* test functions have this property. "Open sourced" refers to open access to most algorithms involved in the published comparison; here, "-" refers to license issues for the benchmark itself.

Testbed	BBOB	MuJoCo	LSGO	Nevergrad	BBComp	OptimSuite
Large scale	-		+	+		+
Translations	+		+	+	+	+
Symmetrizations / rotations	+		+	+		+
One-line reproducibility	-		-	+		+
Periodic automated dashboard				+		+
Complex or real-world	-	+	-	+		+
Multimodal	+	+	+	+	+	+
Open sourced / no license		-				+
Ask/tell/recommend correct	-		+	+	+	+
Far-optimum	+			+		+
Human excluded / client-server					+	

For composite functions which use various objective functions on various subsets of variables, we consider the case with rotations – without excluding the non-rotated case. An extension of symmetrization that we will integrate later in ABBO, which makes sense for replicating an objective function without exact identity, consists in symmetrizing some variables: for example, if the i^{th} variable has range $[a, b]$, we can replace x_i by $b + a - x_i$. Applying this on various subsets of variables leads to 2^d symmetries of an objective function, if the dimension is d. This variation can reduce the bias toward symmetric search operations [LTO+13].

Benchmarking in OptimSuite. We summarize in Table 1 some existing benchmark collections and their (desirable) properties. We inherit various advantages from Nevergrad, namely the automatic rerun of experiments and reproducibility in one line. Our fork includes PBT (a small scale version of Population-Based Training [JDO+17]), Pyomo [HLW+17], Photonics (problems related to optical properties and nanometric materials), YABBOB and variants, LSGO [LTO+13], MLDA [GS18], PowerSystems, FastGames, 007, Rocket, SimpleTSP, Realworld [LMP+20], MuJoCo [TET12], and others, including a (currently small) benchmark of hyperparameters of Scikit-Learn [PVG+11] and Keras-tuning (underlined means: the benchmark is either new, or, in the case of PowerSystems or SimpleTSP, significantly modified compared to previous works, or, in the case of LSGO or MuJoCo, included for the first time inside Nevergrad. For MuJoCo, we believe that interfacing with Nevergrad is particularly useful, to ensure fair comparisons, which rely very much on the precise setup of MuJoCo. We note that, at present, we do not reproduce the extreme black-box nature of [LG17]. Still, by integrating such a wide range of benchmarks in a single open source framework, which, in addition, is periodically re-run, we believe that Nevergrad/OptimSuite provides a significant contribution to benchmarking, and this both for the optimization and the machine learning community, where most of the benchmark suites originate from.

3 A New Algorithm Selection Wizard: ABBO

Black-box optimization is sometimes dominated by evolutionary computation. Evolution strategies [BS02, Bey01, Rec73] have been particularly dominant in the continuous case, in experimental comparisons based on the Black-Box Optimization Benchmark BBOB [HAFR09] or variants thereof. Parallelization advantages [SGZ+16] are particularly appreciated in the machine learning context. However, Differential Evolution [SP97] is a key component of most winning algorithms in competitions based on variants of Large Scale Global Optimization (LSGO [LTO+13]), suggesting a significant difference between these benchmarks. In particular, LSGO is more based on correctly identifying a partial decomposition and scaling to $\geq 1\,000$ variables, whereas BBOB focuses (mostly, except [VAB+18]) on ≤ 40 variables. Mathematical programming techniques [Pow64, Pow94, NM65, Art15] are rarely used in the evolutionary computation world,

Table 2: Nevergrad maintains a dashboard [RT20]. For each experiment, there are many configurations (budget, objective function, possibly dimension, and noise level). We separate benchmarks used for designing ABBO, benchmarks used for validation, and those only used for testing. "*" denotes benchmarks used for designing Shiwa (which is used inside ABBO). We present the rank based on the winning rate of ABBO in the dashboard. Since the submission of this paper, several variants of bandit-based algorithms have been added for high-dimensional noisy optimization. They outperform ABBO, hence its poor rank for these cases. Detailed plots are available in Fig. 4, 7. As expected, DE variants are strong on LSGO and CMA-ES variants are strong for YABBOB. ABBO also performs well on YABBOB, which was used for designing its ancestor Shiwa (see Fig. 1). For the MuJoCo testbed, details are available in Table 3 and Figure 3. Our modifications in the codebase implies an improvement of Shiwa compared to the version published in [LMP+20]; for example, our chaining implies that the $(k+1)^{th}$ code starts from the best point obtained by the k^{th} algorithm, which significantly improves in particular the chaining CMA-ES+Powell or CMA-ES+SQP. Experiments with "†" in the ranking of Shiwa correspond to this improved version of Shiwa.

Benchmark	**Use for ABBO**	**# of configs**	**ranking**			**ABBO best competitor**
			ABBO	Shiwa	CMA-ES	
HDBO	Designing	24	2/21	$1^\dagger$	2	Shiwa
PARAMULTIMODAL	Designing	112	**1/27**	$3^\dagger$	6	DiagonalCMA-ES [RH08]
Realworld	Designing	486	**1/6**	$2^\dagger$	3	Shiwa [LMP+20]
Illcondi	Designing	12	**1/24**	$3^\dagger$	8	Cobyla
Illcondipara	Designing	12	5/28	$7^\dagger$	3	DiagonalCMA-ES
YABBOB	Designing*	630	**1/8**	2	5	Shiwa
YAPARABBOB	Designing*	630	**1/6**	4	5	MetaModel
YAHDBBOB	Designing*	378	2/19	3	18	$(1+1)$-ES
YANOISYBBOB	Designing*	630	2/11	6	10	SQP
YAHDNOISYBBOB	Designing*	630	4/24	2	13	SQP
YASMALLBBOB	Designing*	378	**1/8**	2	7	Shiwa
HdMultimodal	Validation	42	**1/14**	$2^\dagger$	4	Shiwa
Noisy	Validation	96	16/28	$19^\dagger$	NA	RecombiningOptimisticNoisyDiscrete$(1+1)$
RankNoisy	Validation	72	4/25	NA	19	ProgD13
AllDEs	Validation	60	**1/28**	$2^\dagger$	3	Shiwa [LMP+20]
Pyomo	Evaluating	104	**1/19**	$3^\dagger$	10	Shiwa [LMP+20]
Rocket	Evaluating	13	5/18	$4^\dagger$	3	DiagonalCMA-ES [RH08]
SimpleTSP	Evaluating	52	3/15	$2^\dagger$	7	PortfolioDiscrete$(1+1)$
Seq. Fastgames	Evaluating	20	3/28	$4^\dagger$	23	OptimisticDiscrete$(1+1)$
LSGO	Evaluating	45	**1/6**	$4^\dagger$	6	MiniLHSDE
Powersystems	Evaluating	48	10/26	NA	25	$(1+1)$-ES

but they have won competitions [Art15] and significantly improved evolution strategies through memetic methods [RS94]. Algorithm selection was applied to continuous black-box optimization and pushed in Nevergrad [LMP+20]: their optimization algorithm combines many optimization methods and outperforms each of them when averaged over diverse test functions. Closer to machine learning, efficient global optimization [JSW98] is widely used, although it suffers from the curse of dimensionality more than other methods [SLA12]; [WFT20] presented a state-of-the-art algorithm in black-box optimization on MuJoCo, i.e., for the control of various realistic robots [TET12]. We propose ABBO, which extends [LMP+20] by the following features:

(1) Better use of chaining [MLH09] and more intensive use of mathematical programming techniques for the last part of the optimization run, i.e., the local convergence, thanks to Meta-Models (in the parallel case) and more time spent in Powell's method (in the sequential case). This explains the improvement visible in Section 4.1.

(2) Better performance in discrete optimization, using additional codes recently introduced in Nevergrad, in particular adaptive step-sizes.

(3) Better segmentation of the different cases of continuous optimization. We still entirely rely on the base algorithms as available in Nevergrad; that is, we did not modify the tuning of any method. We acknowledge that our method only work thanks to the solid base components available in Nevergrad, which are based on contributions from various research teams.

The obtained algorithm selection wizard, ABBO, is presented in Algorithm 1. The performances of ABBO is summarized in Table 2 and a detailed dashboard is available at `https://dl.fbaipublicfiles.com/nevergrad/allxps/list.html`.

4 Experimental Results

When presenting results on a single benchmark function, we present the usual average objective function value for different budget values. When a collection comprises multiple benchmark problems, we present our aggregated experimental results with two distinct types of plots:

(1) Average normalized objective value for each budget, averaged over all problems. The normalized objective value is the objective value linearly rescaled to $[0, 1]$.

(2) Heatmaps, showing for each pair (x, y) of optimization algorithms the frequency at which Algorithm x outperforms Algorithm y. Algorithms are ranked by average winning frequency. We use red arrows to highlight ABBO.

4.1 Benchmarks in OptimSuite Used for Designing and Validating ABBO

YABBOB (Yet Another Black-Box Optimization Benchmark [RDP$^+$19]), is an adaptation of BBOB [HAFR09], with extensions such as parallelism and noise management. It contains many variants, including noise, parallelism, high-dimension (BBOB was limited to dimension < 50). Several extensions, for the high-dimensional, the parallel or the big budget case, have been developed: we present results in Figures 1 and 4. The high-dimensional one is inspired by [LTO$^+$13], the noisy one is related to the noisy counterpart of BBOB but correctly implements the difference between ask and recommend as discussed in Section 2. The parallel one generalizes YABBOB to settings in which several evaluations can be executed in parallel. Results on PARAMULTIMODAL are presented in Figure 6 (left). In addition, ABBO was run on ILLCONDI & ILLCONDIPARA (ill conditionned functions), HDMULTIMODAL (a multimodal case focusing on high-dimension), NOISY & RANKNOISY (two noisy continuous testbeds), YAWIDEBBOB (a broad range of functions including discrete cases and cases with constraints).

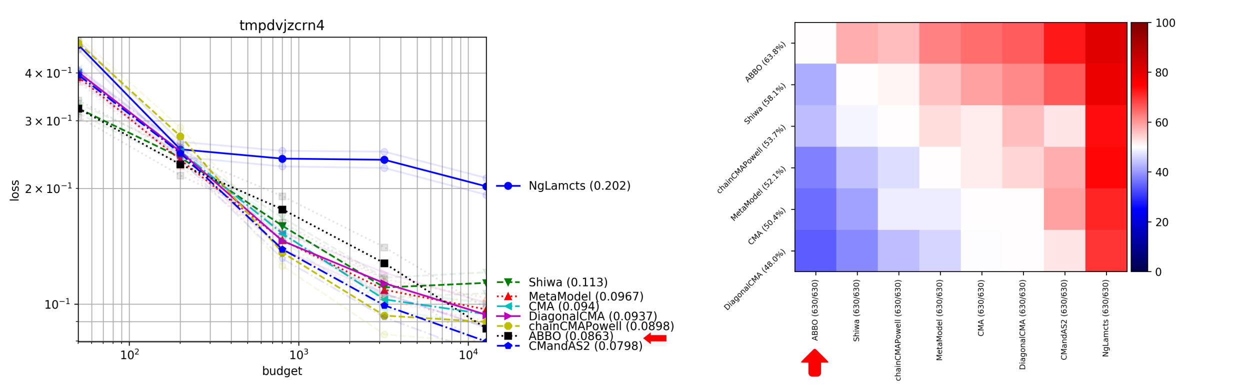

Figure 1: Average normalized loss and heatmap for YABBOB. Additional plots for High-dimensional (HD), NoisyHD, and Large budgets are available in Fig. 4. Other variants include parallel, differences of budgets, and combinations of those variants, with excellent results for ABBO. Up-to-date results from the periodically rerun experiments are available on Nevergrad's dashboard `https://dl.fbaipublicfiles.com/nevergrad/allxps/list.html`, where ABBO is merged under the name NGOpt8.

AllDEs and Hdbo are benchmark collections specifically designed to compare DE variants (AllDEs) and high-dimensional Bayesian Optimization (Hdbo), respectively [RT18]. These benchmark functions are

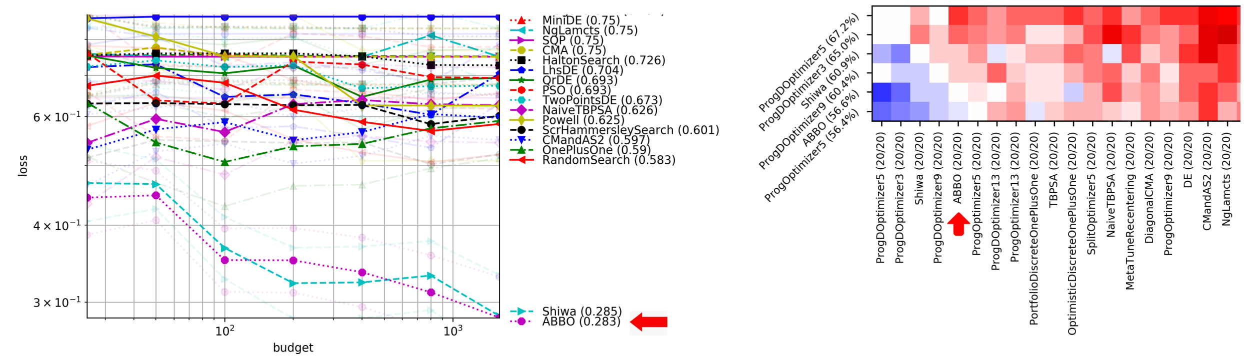

Figure 2: Additional problems: Pyomo (covering Knapsack, P-median and others), SequentialFastgames (presented as heatmaps due to the high noise: GuessWho, War, Batawaf, Flip). Rockets, SimpleTSP, PowerSystems, and LSGO plots are available in Figures 7, and 8. Pyomo and SimpleTSP include discrete variables. Pyomo includes constraints. Rocket, PowerSystems, SequentialFastGames are based on open source simulators and are already merged from OptimSuite to Nevergrad.

similar to the ones used in YABBOB. Many variants of DE (resp. BO) are considered. Results are presented in Figure 5. They show that the performance of ABBO, relatively to DE or BO, is consistent over a wide range of parametrizations of DE or BO, at least in their most classical variants, which are all available in Nevergrad for empirical comparisons.

Realworld: A test of ABBO is performed on the Realworld optimization benchmark suite proposed in [RT18]. This suite includes testbeds from MLDA [GS18] and from [LMP+20]. Results for this suite, presented in Figure 6, confirm that ABBO performs well also on benchmarks that were not explicitly used for its design - however, this benchmark was used for designing Shiwa, which was the basis of our ABBO. A rigorous cross-validation, on benchmarks totally independent from the design of Shiwa, is provided in the next sections.

4.2 New Benchmarks in OptimSuite Used Only for Evaluating ABBO

Pyomo is a modeling language in Python for optimization problems [HLW+17]. It is popular and has been adopted in formulating large models for complex and real-world systems, including energy systems and network resource systems. We implemented an interface to Pyomo for Nevergrad. Experimental results are shown in Figure 2. They show that ABBO also performs decently in discrete settings and in constrained cases.

Additional new artificial and real-world functions: LSGO (large scale global optimization) combines various functions into an aggregated difficult testbed including composite highly multimodal functions. Correctly decomposing the problem is essential. Various implementations of LSGO exist; in particular we believe that some of them do not match exactly. Our implementation follows [LTO+13] , which introduces functions with subcomponents (i.e., groups of decision variables) having non-uniform sizes and non-uniform, even conflicting, contributions to the objective function. Furthermore, we present here experimental results on SequentialFastgames from the Nevergrad benchmarks, and three newly introduced benchmarks, namely Rocket, SimpleTSP (a set of traveling salesman problems), power systems (unit commitment problems [Pad04]). Experimental results are presented in Figures 2, 7, and 8. They show that ABBO performs well on new benchmarks, never used for its design nor for that of the low-level heuristics used inside ABBO.

MuJoCo. Many articles [SK20, WFT20] studied the MuJoCo testbeds [TET12] in the black-box setting. MuJoCo tasks correspond to control problems. Defined in [WFT20, MGR18], the objective is to learn a linear mapping from states to actions. It turned out that the scaling is critical [MGR18]: for reasons mentioned

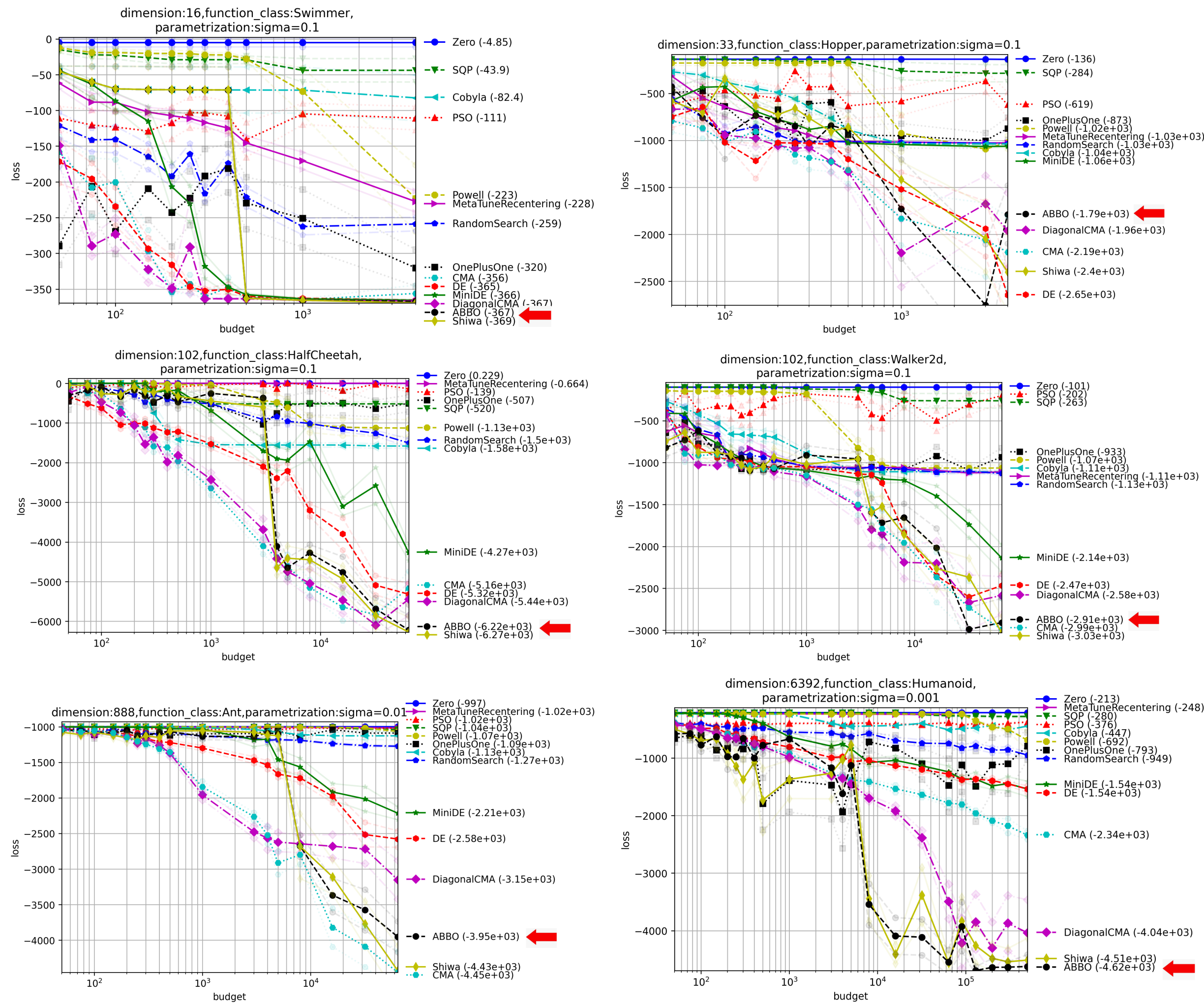

Figure 3: Results on the MuJoCo testbeds. Dashed lines show the standard deviation. Compared to the state of the art in [WFT20], with an algorithm adapted manually for the different tasks, we get overall better results for Humanoid, Ant, Walker. We get worse results for Swimmer (could match if we had modified our code for the 3 easier tasks as done in [WFT20]), similar for Hopper and Cheetah: we reach the target for 5 of the 6 problems (see text). Runs of Shiwa correspond to the improvement of Shiwa due to chaining, as explained in Table 2.

in Section 2, solutions are close to 0. We chose to scale all the variables of the problem at the power of 0.1 the closest to $1.2/d$, for all methods run in Figure 3. We remark that ABBO and Shiwa perform well, including comparatively to gradient-based methods in some cases, while having the ability to work when the gradient is not available. When the gradient is available, black-box methods do not require computation of the gradient, which saves time.

We use the same experimental setup as [WFT20] (linear policy, offline whitening of states). We get results better than LA-MCTS, in a setting i.e., does not use any expensive surrogate model (Table 3). Our runs with CMA-ES and Shiwa are better than those in [WFT20]. We acknowledge that LMRS [SK20] outperforms our method on all MuJoCo tasks, using a deep network as a surrogate model: however, we point out that a part of their code is not open sourced, making the experiments not reproducible. In addition, when rerunning their repository without the non open sourced part, it solved Half-Cheetah within budget 56k, which is larger

Table 3: Results for a linear policy in the black-box setting from the latest black-box paper [WFT20] and references therein, compared to results from ABBO. Two last columns = average reward for the maximum budget tested in [WFT20], namely 1k, 4k, 4k, 40k, 30k, 40k, respectively. "ioa" = iterations on average for reaching the target. "iter" = iterations for target reached for median run. "*" refers to problems for which the target was not reached by [WFT20]: then BR means "best result in 10 runs". ABBO reaches the target for Humanoid and Ant whereas previous (black-box) papers did not; we get nearly the same ioa for Hopper and HalfCheetah (Nevergrad computed the expected value instead of computing the ioa, so we cannot compare exactly; see Figure 3 for curves). ABBO is slower than LA-MCTS on Swimmer. Note that we keep the same method for all benchmarks whereas LA-MCTS modified the algorithm for 3 rows. On HDMULTIMODAL, ABBO performs better than LA-MCTS, as detailed in the text, and as confirmed in [WFT20], which acknowledges the poor results of LA-MCTS for high-dimensional Ackley and Rosenbrock.

Task	Target	LA-MCTS results	ABBO result	LA-MCTS avg reward	ABBO avg reward
Swimmer-v2	325	132 ioa	around 450 iter	358	**365**
Hopper-v2	3120	2897 ioa	around 3 000 iter	**3292**	1787
HalfCheetah-v2	3430	3877 ioa	around 4 000 iter	3227	**4730**
Walker2d-v2*	4390	BR: 3314 (not reached)	BR: **4398, budget < 64 000 (reached!)**	2769	**2949**
Ant-v2*	3580	BR: 2791 (not reached)	BR: **5325, budget < 32 000 (reached!)**	2511	**3532**
Humanoid-v2*	6 000	BR: 3384 (not reached)	BR (budget 5 00000): **4870**	2511	**4620**

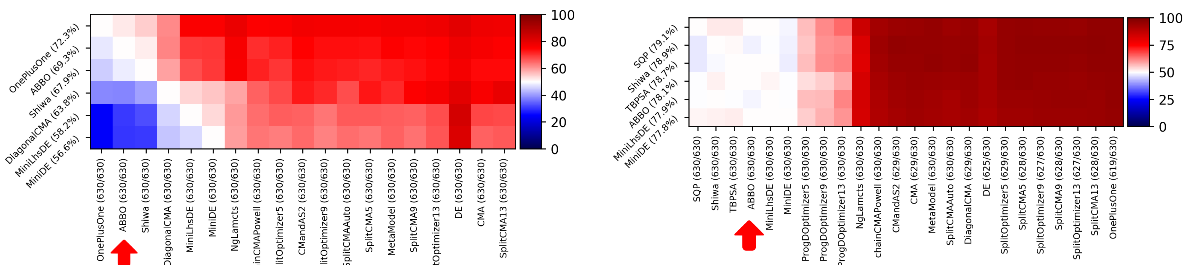

Figure 4: YAHDBBOB (dimension ≥ 50) and YANOISYHDBBOB (noisy + dimension ≥ 50) heatmaps.

than ours. For Humanoid, the target was reached at 768k, which is again larger than our budget. Results from ABBO are comparable to, and usually better than (for the 3 hardest problems), results from LA-MCTS, while ABBO is entirely reproducible. In addition, it runs the same method for all benchmarks and it is not optimized for each task specifically as in [SK20, WFT20]. In contrast to ABBO, [WFT20] uses different underlying regression methods and sampling methods depending on the MuJoCo task, and it is not run on other benchmarks except for some of the HDMULTIMODAL ones. On the latter, ABBO performances are significantly better for Ackley and Rosenbrock in dimension 100 (expected results around 100 and 10^{-8} after 10k iterations for Rosenbrock and Ackley respectively for ABBO, vs 0.5 and 500 in [WFT20]). From the curves in [WFT20] and in the present work, we expect LA-MCTS to perform well with an adapted choice of parametrization and with a low budget, for tasks related to MuJoCo, whereas ABBO is adapted for wide ranges of tasks and budgets.

5 Conclusions

This paper proposes OptimSuite, a very broad benchmark suite composed of real-world and of artificial benchmark problems. OptimSuite is implemented as a fork of Nevergrad [RT18], from which it inherits a strong reproducibility: our (Python) code is open source, tests are rerun periodically, and up-to-date results are available in the public dashboard [RT20]. A whole experiment can be done as a one-liner. OptimSuite fixes several issues of existing benchmarking environments. The suite subsumes MuJoCo, Pyomo, LSGO, YABBOB, MLDA, and several new real-world problems. We also propose ABBO, an improved

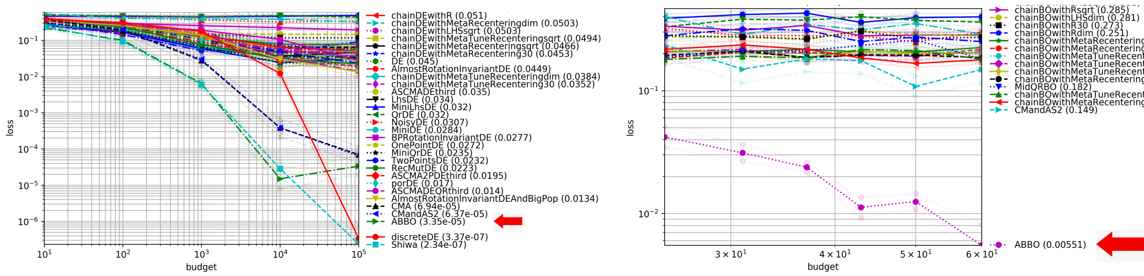

Figure 5: ABBO vs specific families of optimization algorithms (DE, and BO in the high-dimensional case) on Cigar, Hm, Ellipsoid, Sphere functions. Not all run algorithms are mentioned, for short. Bayesian optimization (Nevergrad uses [Nog]), often exploring boundaries first, is outperformed in high dimension [WFT20].

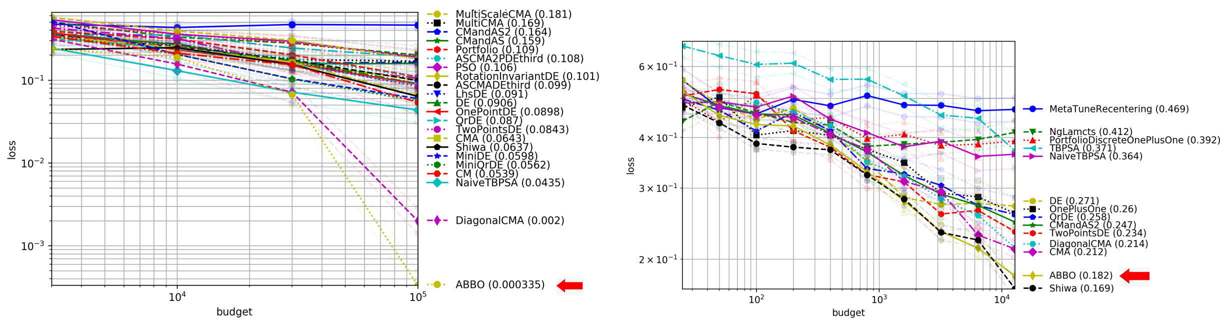

Figure 6: Up: experiments for the parallel multimodal setting PARAMULTIMODAL. Budget up to 100 000, parallelism 1 000, Ackley+Rosenbrock+DeceptiveMultimodal+Griewank+Lunacek+Hm. Bottom: Real-world benchmark from Nevergrad: games, Sammon mappings, clustering, small traveling salesman instance, small power systems.

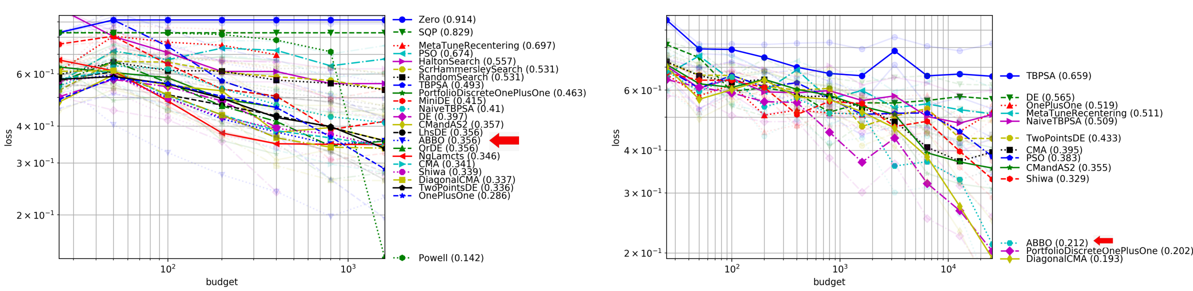

Figure 7: Additional problems (1): on left, Rocket (26 continuous variables, budget up to 1600, sequential or parallelism 30) and on right, SimpleTSP (10 to 1 000 decision variables).

algorithm selection wizard. Despite its simplicity, ABBO shows very promising performance across the whole benchmark suite, often outperforming the previous state-of-the-art, problem-specific solvers: (a) by solving 5 of the 6 cases without any task-specific hyperparameter tuning, ABBO outperforms LA-MCTS, which was specialized for each single task, (b) ABBO outperforms Shiwa on YABBOB and its variants, which is the benchmark suite used to design Shiwa in the first place, (c) ABBO is also among the best methods on LSGO and almost all other benchmarks.

Future work. OptimSuite subsumes most of the desirable features outlined in Section 2, with one notable exception, the true black-box setting, which other benchmark environments have implemented through a

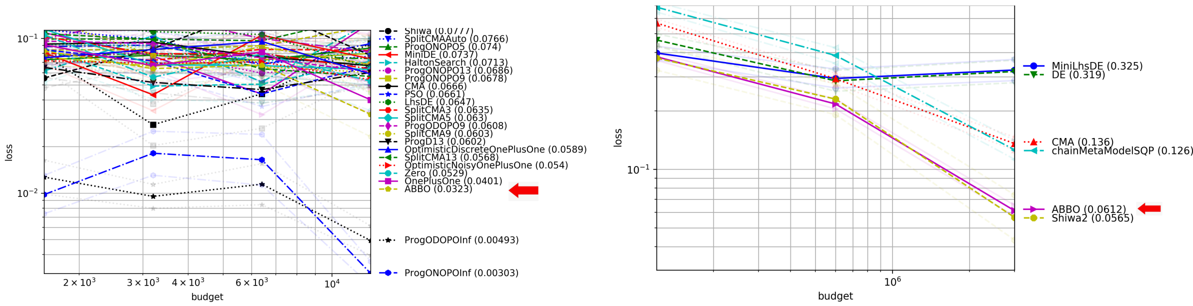

Figure 8: Additional problems (2): on left, PowerSystems (1806 to 9646 neural decision variables) and on right, LSGO (mix of partially separable, overlapping, shifted cases as in [LTO+13]).

client-server interaction [LG17]. A possible combination between our platform and such a challenge, using the dashboard to publish the results, could be useful, to offer a meaningful way for cross-validation. Further improving ABBO is on the roadmap. In particular, we are experimenting with the automation of the still hand-crafted selection rules. Note, though, that it is important to us to maintain a high level of interpretability, which we consider key for a wide acceptance of the wizard. Another avenue for future work is a proper configuration of the low-level heuristics subsumed by ABBO. At present, some of them are merely textbook implementations, and significant room for improvement can therefore be expected. Newer variants [Los14, AH16, LGB18] of CMA-ES, of LMRS [SK20], recent Bayesian optimization libraries (e.g. [EPG+19]), as well as per-instance algorithm configuration such as [BDSS17] are not unlikely to result in important improvements for various benchmarks. We also plan on extending OptimSuite further, both through interfacing existing benchmark collections/problems, and by designing new benchmark problems ourselves.

Acknowledgments

This work was supported by the Paris Ile-de-France Region.

References

[AH09] Anne Auger and Nikolaus Hansen, *Benchmarking the (1+1)-CMA-ES on the BBOB-2009 noisy testbed*, Proc. of Genetic and Evolutionary Computation Conference (GECCO'09, Companion Material, ACM, 2009, pp. 2467–2472.

[AH16] Youhei Akimoto and Nikolaus Hansen, *Projection-based restricted covariance matrix adaptation for high dimension*, Proc. of Genetic and Evolutionary Computation (GECCO'16), 2016, pp. 197–204.

[Art15] SME Artelys, *Artelys sqp wins the bbcomp competition*, 2015.

[AST05] Anne Auger, Marc Schoenauer, and Olivier Teytaud, *Local and global order 3/2 convergence of a surrogate evolutionary algorithm*, Proc. of Genetic and Evolutionary Computation Conference (GECCO'05), ACM, 2005, pp. 857–864.

[BDSS17] Nacim Belkhir, Johann Dréo, Pierre Savéant, and Marc Schoenauer, *Per instance algorithm configuration of cma-es with limited budget*, Proc. of Genetic and Evolutionary Computation Conference (GECCO'17), ACM, 2017, p. 681–688.

[Ber16] Vincent Berthier, *Progressive differential evolution on clustering real world problems*, Revised Selected Papers of the 12th International Conference on Artificial Evolution - Volume 9554, Springer-Verlag New York, Inc., 2016, pp. 71–82.

[Bey01] Hans-Georg Beyer, *The theory of evolution strategies*, Natural Computing Series, Springer, Heideberg, 2001.

[Bey12a] ———, `http://lists.lri.fr/pipermail/bbob-discuss/2012-April/000270.html`, 2012.

[Bey12b] ———, `http://lists.lri.fr/pipermail/bbob-discuss/2012-April/000258.html`, 2012.

[BGK+17] Olivier Bousquet, Sylvain Gelly, Kurach Karol, Olivier Teytaud, and Damien Vincent, *Critical hyper-parameters: No random, no cry*, CoRR **abs/1706.03200** (2017).

[BKK+16] Bernd Bischl, Pascal Kerschke, Lars Kotthoff, Thomas Marius Lindauer, Yuri Malitsky, Alexandre Fréchette, Holger H. Hoos, Frank Hutter, Kevin Leyton-Brown, Kevin Tierney, and Joaquin Vanschoren, *ASlib: A Benchmark Library for Algorithm Selection*, Artificial Intelligence (AIJ) **237** (2016), 41 – 58.

[BMS11] Sébastien Bubeck, Rémi Munos, and Gilles Stoltz, *Pure exploration in finitely-armed and continuous-armed bandits*, Theor. Comput. Sci. **412** (2011), no. 19, 1832–1852.

[BS02] Hans-Georg Beyer and Hans-Paul Schwefel, *Evolution Strategies - A Comprehensive Introduction*, Natural Computing **1** (2002), no. 1, 3–52.

[BS04] Nicolas Baskiotis and Michèle Sebag, *C4.5 competence map: a phase transition-inspired approach*, Machine Learning, Proceedings of the Twenty-first International Conference (ICML 2004), Banff, Alberta, Canada, July 4-8, 2004, 2004.

[CAH12] Alexandre Adrien Chotard, Anne Auger, and Nikolaus Hansen, *Cumulative Step-size Adaptation on Linear Functions: Technical Report*, Research report, Inria Saclay, June 2012.

[CCD+19] Marie-Liesse Cauwet, Camille Couprie, Julien Dehos, Pauline Luc, Jérémy Rapin, Morgane Riviere, Fabien Teytaud, and Olivier Teytaud, *Fully parallel hyperparameter search: Reshaped space-filling*, arXiv preprint arXiv:1910.08406. To appear in Proc. of ICML 2020 (2019).

[CHP+10] Yann Collette, Nikolaus Hansen, Gilles Pujol, Daniel Salazar, and Rodolphe Le Riche, *On object-oriented programming of optimizers - examples in scilab.*

[CLRT16] Marie-Liesse Cauwet, Jialin Liu, Baptiste Rozière, and Olivier Teytaud, *Algorithm portfolios for noisy optimization*, Annals of Mathematics and Artificial Intelligence **76** (2016), no. 1-2, 143–172.

[Cou12a] Remi Coulom, `http://lists.lri.fr/pipermail/bbob-discuss/2012-April/000252.html`, 2012.

[Cou12b] Rémi Coulom, *Clop: Confident local optimization for noisyblack-box parameter tuning*, Advances in Computer Games, Springer Berlin Heidelberg, 2012, pp. 146–157.

[CT20] Marie-Liesse Cauwet and Olivier Teytaud, *Population control meets Doob's martingale theorems: The noise-free multimodal case*, Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion, GECCO '20, Association for Computing Machinery, 2020, p. 321–322.

[DDL19] Benjamin Doerr, Carola Doerr, and Johannes Lengler, *Self-adjusting mutation rates with provably optimal success rules*, Proceedings of the Genetic and Evolutionary Computation Conference, GECCO '19, Association for Computing Machinery, 2019, p. 1479–1487.

[DL16] Duc-Cuong Dang and Per Kristian Lehre, *Self-adaptation of mutation rates in non-elitist populations*, Parallel Problem Solving from Nature - PPSN XIV - 14th International Conference, 2016, pp. 803–813.

[DLMN17] Benjamin Doerr, Huu Phuoc Le, Régis Makhmara, and Ta Duy Nguyen, *Fast genetic algorithms*, Proceedings of the Genetic and Evolutionary Computation Conference, GECCO '17, ACM, 2017, pp. 777–784.

[DT13] Jérémie Decock and Olivier Teytaud, *Noisy optimization complexity under locality assumption*, Proceedings of the Twelfth Workshop on Foundations of Genetic Algorithms XII, FOGA XII '13, ACM, 2013, pp. 183–190.

[EPG+19] David Eriksson, Michael Pearce, Jacob Gardner, Ryan D Turner, and Matthias Poloczek, *Scalable global optimization via local Bayesian optimization*, Advances in Neural Information Processing Systems (H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alché-Buc, E. Fox, and R. Garnett, eds.), vol. 32, Curran Associates, 2019, pp. 5496–5507.

[FM14] Matteo Fischetti and Michele Monaci, *Exploiting erraticism in search*, Operations Research **62** (2014), no. 1, 114–122.

[GS18] Marcus Gallagher and Sobia Saleem, *Exploratory landscape analysis of the mlda problem set*, PPSN'18 workshop, 2018.

[HAFR09] Nikolaus Hansen, Anne Auger, Steffen Finck, and Raymond Ros, *Real-parameter black-box optimization benchmarking 2009: Experimental setup*, Tech. Report RR-6828, INRIA, France, 2009.

[HB16] Michael Hellwig and Hans-Georg Beyer, *Evolution under strong noise: A self-adaptive evolution strategy can reach the lower performance bound - the pcCMSA-ES*, Parallel Problem Solving from Nature – PPSN XIV, Springer International Publishing, 2016, pp. 26–36.

[HLW+17] William E Hart, Carl D Laird, Jean-Paul Watson, David L Woodruff, Gabriel A Hackebeil, Bethany L Nicholson, and John D Siirola, *Pyomo-optimization modeling in python*, vol. 67, Springer, 2017.

[HMI09] Verena Heidrich-Meisner and Christian Igel, *Hoeffding and bernstein races for selecting policies in evolutionary direct policy search*, Proceedings of the 26th Annual International Conference on Machine Learning, ICML '09, ACM, 2009, pp. 401–408.

[HO03] Nikolaus Hansen and Andreas Ostermeier, *Completely derandomized self-adaptation in evolution strategies*, Evolutionary Computation **11** (2003), no. 1.

[Hol75] John H. Holland, *Adaptation in natural and artificial systems*, University of Michigan Press, 1975.

[HRM+11] Nikolaus Hansen, Raymond Ros, Nikolas Mauny, Marc Schoenauer, and Anne Auger, *Impacts of Invariance in Search: When CMA-ES and PSO Face Ill-Conditioned and Non-Separable Problems*, Applied Soft Computing **11** (2011), 5755–5769.

[JDO+17] Max Jaderberg, Valentin Dalibard, Simon Osindero, Wojciech M. Czarnecki, Jeff Donahue, Ali Razavi, Oriol Vinyals, Tim Green, Iain Dunning, Karen Simonyan, Chrisantha Fernando, and Koray Kavukcuoglu, *Population based training of neural networks*, 2017.

[JSW98] Donald R. Jones, Matthias Schonlau, and William J. Welch, *Efficient global optimization of expensive black-box functions*, Journal of Global Optimization **13** (1998), no. 4, 455–492.

[KHNT18] Pascal Kerschke, Holger H. Hoos, Frank Neumann, and Heike Trautmann, *Automated Algorithm Selection: Survey and Perspectives*, Evolutionary Computation (2018), 1 – 47.

[Kot14] Lars Kotthoff, *Algorithm Selection for Combinatorial Search Problems: A Survey*, AI Magazine **35** (2014), no. 3, 48 – 60.

[KT18] Pascal Kerschke and Heike Trautmann, *Automated Algorithm Selection on Continuous Black-Box Problems By Combining Exploratory Landscape Analysis and Machine Learning*, Evolutionary Computation (2018), 1 – 28.

[LG17] Ilya Loshchilov and T. Glasmachers, *Black box optimization competition*, 2017.

[LGB18] Ilya Loshchilov, Tobias Glasmachers, and Hans-Georg Beyer, *Large scale black-box optimization by limited-memory matrix adaptation*, IEEE Transactions on Evolutionary Computation **23** (2018), no. 2, 353–358.

[LMP+20] Jialin Liu, Antoine Moreau, Mike Preuss, Jeremy Rapin, Baptiste Roziere, Fabien Teytaud, and Olivier Teytaud, *Versatile black-box optimization*, Proceedings of the 2020 Genetic and Evolutionary Computation Conference, GECCO '20, 2020, p. 620–628.

[Los14] Ilya Loshchilov, *A computationally efficient limited memory cma-es for large scale optimization*, Proc. of Genetic and Evolutionary Computation (GECCO'14), 2014, pp. 397–404.

[LTO+13] Xiaodong Li, Ke Tang, Mohammmad Nabi Omidvar, Zhenyu Yang, and Kai Qin, *Benchmark functions for the CEC'2013 special session and competition on large-scale global optimization.*

[MBT+11] Olaf Mersmann, Bernd Bischl, Heike Trautmann, Mike Preuss, Claus Weihs, and Günter Rudolph, *Exploratory Landscape Analysis*, Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation (GECCO), ACM, 2011, pp. 829 – 836.

[MDRT20] Laurent Meunier, Carola Doerr, Jérémy Rapin, and Olivier Teytaud, *Variance reduction for better sampling in continuous domains*, Parallel Problem Solving from Nature - PPSN XVI - 16th International Conference, PPSN 2020, Leiden, The Netherlands, September 5-9, 2020, Proceedings, Part I, Lecture Notes in Computer Science, vol. 12269, Springer, 2020, pp. 154–168.

[ME13] Katherine Mary Malan and Andries Petrus Engelbrecht, *A Survey of Techniques for Characterising Fitness Landscapes and Some Possible Ways Forward*, Information Sciences (JIS) **241** (2013), 148 – 163.

[MGR18] Horia Mania, Aurelia Guy, and Benjamin Recht, *Simple random search provides a competitive approach to reinforcement learning*, 2018.

[MLH09] D. Molina, M. Lozano, and F. Herrera, *Memetic algorithm with local search chaining for continuous optimization problems: A scalability test*, 2009 Ninth International Conference on Intelligent Systems Design and Applications, 2009, pp. 1068–1073.

[MSKH15] Mario Andrés Muñoz Acosta, Yuan Sun, Michael Kirley, and Saman K. Halgamuge, *Algorithm Selection for Black-Box Continuous Optimization Problems: A Survey on Methods and Challenges*, Information Sciences (JIS) **317** (2015), 224 – 245.

[NM65] John A. Nelder and Roger Mead, *A simplex method for function minimization*, Computer Journal **7** (1965), 308–313.

[Nog] Fernando Nogueira, *Bayesian Optimization: Open source constrained global optimization tool for Python*, 2014–.

[PA12] Erik Pitzer and Michael Affenzeller, *A Comprehensive Survey on Fitness Landscape Analysis*, Recent Advances in Intelligent Engineering Systems (János Fodor, Ryszard Klempous, and Carmen Paz Suárez Araujo, eds.), Studies in Computational Intelligence, Springer, 2012, pp. 161 – 191.

[Pad04] Narayana Prasad Padhy, *Unit commitment-a bibliographical survey*, IEEE Transactions on Power Systems **19** (2004), no. 2, 1196–1205.

[Pow64] Michael J.D. Powell, *An efficient method for finding the minimum of a function of several variables without calculating derivatives*, The Computer Journal **7** (1964), no. 2, 155–162.

[Pow94] ———, *A direct search optimization method that models the objective and constraint functions by linear interpolation*, pp. 51–67, Springer Netherlands, 1994.

[PVG+11] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay, *Scikit-learn: Machine learning in Python*, Journal of Machine Learning Research **12** (2011), 2825–2830.

[RDP+19] Jérémy Rapin, Pauline Dorval, Jules Pondard, Nicolas Vasilache, Marie-Liesse Cauwet, Camille Couprie, and Olivier Teytaud, *Openly revisiting derivative-free optimization*, Proceedings of the Genetic and Evolutionary Computation Conference Companion, GECCO 2019, Prague, Czech Republic, July 13-17, 2019, ACM, 2019, pp. 267–268.

[Rec73] Ingo Rechenberg, *Evolutionstrategie: Optimierung technischer systeme nach prinzipien des biologischen evolution*, Fromman-Holzboog Verlag, 1973.

[RH08] Raymond Ros and Nikolaus Hansen, *A simple modification in CMA-ES achieving linear time and space complexity*, Parallel Problem Solving from Nature – PPSN X, Springer Berlin Heidelberg, 2008, pp. 296–305.

[Ric76] John Rischard Rice, *The Algorithm Selection Problem*, Advances in Computers **15** (1976), 65 – 118.

[Rog87] Hartley Rogers, *Theory of recursive functions and effective computability*, MIT Press, 1987.

[RRSS18] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar, *Do CIFAR-10 classifiers generalize to CIFAR-10 ?*, 2018.

[RS94] Nicholas J. Radcliffe and Patrick D. Surry, *Formal memetic algorithms*, Evolutionary Computing: AISB Workshop (T.C. Fogarty, ed.), Springer Verlag LNCS 865, 1994, pp. 1–16.

[RT18] Jeremy Rapin and Olivier Teytaud, *Nevergrad - A gradient-free optimization platform*, `https://GitHub.com/FacebookResearch/Nevergrad`, 2018.

[RT20] ———, *Dashboard of results for Nevergrad platform*, `https://dl.fbaipublicfiles.com/nevergrad/allxps/list.html`, 2020.

[Sal96] Ralf Salomon, *Re-evaluating genetic algorithm performance under coordinate rotation of benchmark functions. a survey of some theoretical and practical aspects of genetic algorithms*, BioSystems **39** (1996), no. 3, 263–278.

[SGZ+16] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen, *Improved techniques for training gans*, NeurIPS, 2016.

[SK20] Ozan Sener and Vladlen Koltun, *Learning to guide random search*, International Conference on Learning Representations, 2020.

[SLA12] Jasper Snoek, Hugo Larochelle, and Ryan P. Adams, *Practical bayesian optimization of machine learning algorithms*, Advances in Neural Information Processing Systems (NIPS), 2012, pp. 2951–2959.

[Smi09] Kate Amanda Smith-Miles, *Cross-Disciplinary Perspectives on Meta-Learning for Algorithm Selection*, ACM Computing Surveys (CSUR) **41** (2009), 1 – 25.

[SP97] Rainer Storn and Kenneth Price, *Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces*, J. of Global Optimization **11** (1997), no. 4, 341–359.

[TET12] Emanuel Todorov, Tom Erez, and Yuval Tassa, *Mujoco: A physics engine for model-based control*, 10 2012, pp. 5026–5033.

[VAB+18] Konstantinos Varelas, Anne Auger, Dimo Brockhoff, Nikolaus Hansen, Ouassim Ait ElHara, Yann Semet, Rami Kassab, and Frédéric Barbaresco, *A comparative study of large-scale variants of CMA-ES*, Proc. of Parallel Problem Solving from Nature (PPSN XV), LNCS, vol. 11101, Springer, 2018, pp. 3–15.

[VGO+19] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, CJ Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake Vand erPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1. 0 Contributors, *SciPy 1.0–Fundamental Algorithms for Scientific Computing in Python*, arXiv e-prints (2019), arXiv:1907.10121.

[VHCM15] Mauro Vallati, Frank Hutter, Lukás Chrpa, and Thomas Leo McCluskey, *On the effective configuration of planning domain models*, Proc. of International Joint Conference on Artificial Intelligence (IJCAI'15), 2015.

[WFT20] Linnan Wang, Rodrigo Fonseca, and Yuandong Tian, *Learning search space partition for black-box optimization using Monte Carlo Tree Search*, arXiv:2007.00708. To appear in Proc. of Neurips 2020 (2020).

[XHHLB08] Lin Xu, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown, *SATzilla: Portfolio-based algorithm selection for SAT*, J. Artif. Int. Res. **32** (2008), no. 1, 565–606.

A APPENDIX

We specify the properties mentioned in Table 1.

- **Large scale:** includes dimension $\geq 1\,000$.
- **Translations:** in unbounded continuous domains, a standard deviation σ has to be provided, for example for sampling the first and second iterates of the optimization algorithm. Given a standard deviation σ, we consider that there is translation when optima are randomly translated by a $\mathcal{N}(0,\sigma^2)$ shift. Only interesting for artificial cases.
- **Far-optimum:** optima are translated far from the optimum, with standard deviation at least $\mathcal{N}(0, 25\times\sigma^2)$.
- **Symmetrizations / rotations (here assuming an optimum, up to translation, in** 0**)**. Rotation: with a random rotation matrix M, the function $x \mapsto f(x)$ is replaced by $x \mapsto f(M(x))$. Symmetrization: $x \mapsto f(x)$ can be replaced by $x \mapsto f(S(x))$, with S a diagonal matrix with each diagonal coefficient equal to 1 or -1 with probability 50%. We do not request all benchmarks to be rotated: it might be preferable to have both cases considered.
- **One-line reproducibility:** Where reproducibility requires significant coding, it is unlikely to be of great use outside of a very small set of specialists. One-line reproducibility is given when the effort to reproduce an entire experiment does not require more than the execution of a single line. We consider this to be an important feature.
- **Periodic automated dashboard:** are algorithms re-run periodically on new problem instances? Some platforms do not collect the algorithms, and reproducibility is hence not given. An automated dashboard is convenient also because new problems can be added "on the go" without causing problems, as all algorithms will be executed on all these new problem instances. This feature addresses what we consider to be one of the biggest bottlenecks in the current benchmarking environments.
- **Complex or real-world:** Real-world is self-explanatory; complex means a benchmark involving a complex simulator, even if it is not real world. MuJoCo is in the "complex" category.
- **Multimodal:** whether the suite contains problems for which there are local optima which are not global optima.
- **Open sourced / no license:** Are algorithms and benchmarks available under an open source agreement? BBOB does not collect algorithms, MuJoCo requires a license, LSGO and BBOB are not realworld, Mujoco requires a license, BBComp is no longer maintained, Nevergrad does not include complex ML problems without license issue before our work: some people have already applied Nevergrad to MuJoCo, but with our work MuJoCo becomes part of Nevergrad so that people can upload their code in Nevergrad and it will be run on all benchmarks, including MuJoCo.
- **Ask/tell/recommend correctly implemented** [CHP+10, BMS11]: The ask and tell idea (developped in [CHP+10]) is that an optimization algorithm should not come under the format $Optimizer.minimize(objective - function)$ because there are many settings in which this is not possible: you might think of agents optimizing concurrently their own part of an objective function, and problems of reentrance, or asynchronicity. All settings can be recovered from an ask/tell optimization method. This becomes widely used. However, as well known in the bandit literature (you can think of pure exploration bandits [BMS11]), it is necessary to distinguish ask, tell and recommend: the "recommend" method is the one which proposes an approximation of the optimum. Let us develop an example explaining why this matters: the domain is $\{1, 2, 3, 4\}$, and we have a budget of 20 in a noisy case. NoisyBBOB assumes that the optimum is found when "ask" returns the optimum arm: then, the status remains "found" even if the algorithm has no idea where is the optimum and never

comes back nearby. So an algorithm which just iteratively "asks" $1, 2, 3, 4, 1, 2, 3, 4, \ldots$ reaches the optimum in at most 4 iterations. This does not mean anything in the noisy case, as the challenge is to figure out which of the four numbers is the optimum. With a proper ask/tell/recommend, the optimizer chooses an arm at the end of the budget. A simple regret is then computed. Actually this also matters in the noise-free case, but the issue is much more critical in noisy optimization. The case of continuous noisy optimization also has counter-examples and all the best noisy optimization algorithms use ask/tell/recommend. We add the reference to the paper above.

- **Human excluded / client-server:** The problem instances are truly black-box. Algorithms can only suggest points and observe function values, but neither the algorithm nor its designer have access to any other information about the problem apart from the number of variables, their type, ranges, and order. It is impossible to repeat experiments for tuning hyperparameters without "paying" the budget of the HP tuning. This is something we could not do, as everything is public and open sourced: however, we believe that we mitigate this issue by considering a large number of benchmarks.